What Is Claimed Is:

1. A method comprising:

intercepting a call from an executing program to a library function, wherein said function call requests writing of a data block to the heap section of a memory;

determining whether performing said write request would smash the heap;

executing an error handling procedure instead of writing the data block if performing said write request would smash the heap; and

causing the data block to be written as requested if performing said write request would not smash the heap.

2. The method of claim 1, wherein if writing the data block as requested would overflow a buffer in the heap, then the conclusion of said determining step is that performing the write request would smash the heap.

3. The method of claim 1, wherein a parameter accompanying said function call is a start address of a destination memory section where the data block is requested to be written, and wherein determining whether performing said write request would smash the heap comprises:

determining the size of the data block to be written;

determining whether the destination start address is located within a currently allocated buffer in the heap;

concluding that performing said write request would smash the heap if the destination start address is not within any currently allocated buffer; and

concluding that performing said write request would smash the heap if the destination start address is within an identified buffer and said data block's size is greater than the size of the memory section extending from the destination start address to the end of said identified buffer.

4. The method of claim 3, wherein each currently allocated buffer contains a meta-data field, and wherein determining whether the destination start address is within a currently allocated buffer comprises:

searching the heap in one direction for a meta-data field, wherein the search begins at the destination start address; and

determining that the destination start address is not within any currently allocated buffer if the search reaches a heap boundary without finding a valid meta-data field.

5. The method of claim 4, wherein a potential meta-data field is identified by finding a predefined marker in a memory location being examined during the search.

6. The method of claim 4, wherein each currently allocated buffer is associated with an entry in a buffer management table, and wherein a section of memory is identified as a meta-data field only if that memory section contains a pointer to an entry in the buffer management table.

7. The method of claim 6, wherein the memory section is confirmed to be a meta-data field only if the buffer management table entry pointed to also contains a pointer to an address of a memory location in said meta-data field.

8. The method of claim 4, wherein the search for a valid meta-data filed assumes that each memory buffer is aligned at a boundary that is proportional to its size.

9. The method of claim 1, wherein said step of intercepting a function call is provided for by a dynamic link loader resolving to the fault containment wrapper any references to the function in the executing program.

10. The method of claim 1, wherein causing the data to be written as requested comprises:

determining whether the function was previously called;

resolving the called function through an interface function of the dynamic link loader, if the function was not previously called; and

returning without calling the function again if the function was previously called.

11. An article of manufacture comprising a computer-readable medium having stored thereon instructions which instructions comprise a fault-containment wrapper module for library function calls, wherein said instructions when executed cause the processor to:

receive an intercepted call from an executing program to a library function, wherein said function call requests writing of a data block to the heap section of a memory;

determine whether performing said write request would smash the heap;

execute an error handling procedure instead of writing the data block if performing said write request would smash the heap; and

cause the data block to be written as requested if performing said write request would not smash the heap.

12. The article of manufacture of claim 11, wherein said determining step concludes that writing the data block as requested would overflow a buffer in the heap if the instructions determine that performing the write request would smash the heap.

13. The article of manufacture of claim 11, wherein one parameter accompanying said function call is a start address of the memory section where the data block is requested to be written, and wherein the instructions determine whether performing said write request would smash the heap by:

determining the size of the data block to be written;

determining whether a destination start address is located within a currently allocated buffer in the heap; and

concluding that performing said write request would smash the heap if the destination start address is not within any currently allocated buffer; and

concluding that performing said write request would smash the heap if the destination start address is within an identified buffer and said data block's size is greater than the size of the memory section extending from the destination start address to the end of said identified.

14. The article of manufacture of claim 13, wherein each currently allocated buffer contains a meta-data field, and wherein said determining whether the destination start address is within a currently allocated buffer comprises:

searching the heap in one direction for a meta-data field, wherein the search begins at the destination start address; and

determining that the destination start address is not within any currently allocated buffer if the search reaches a heap boundary without finding a valid meta-data field.

15. The article of manufacture of claim 14, wherein a potential meta-data field is identified by finding a predefined marker in a memory location being examined during the search.

16. The article of manufacture of claim 14, wherein each currently allocated buffer is associated with an entry in a buffer management table, and wherein a section of memory is identified as a meta-data field only if that memory section contains a pointer to an entry in the buffer management table.

17. The article of manufacture of claim 16, wherein the memory section is confirmed to be a meta-data field only if the buffer management table entry pointed to also contains a pointer to an address of a memory location in said meta-data field.

18. The article of manufacture of claim 14, wherein the instructions to search for a valid meta-data filed assume that each memory buffer is aligned at a boundary that is proportional to its size.

19. The article of manufacture of claim 14, wherein intercepting of the function call is provided for by a dynamic link loader resolving to the fault containment wrapper any references to the function in the executing program

20. The article of manufacture of claim 14, wherein the instructions cause the data to be written by:

determining whether the function was previously called;

resolving the called function through an interface function of the dynamic link loader if the function was not previously called; and

returning without calling the function again if the function was previously called.

21. A method comprising:

intercepting a call to a library function, wherein the function call provides for writing a data string to a block of addresses within a heap section of a memory; and

initiating a fault-containment wrapper to perform steps that comprise:

determining whether writing the data string to the block of addresses would overflow a buffer within the heap;

causing the data string to be written to the block of addresses if it was determined that said writing would not overflow a buffer; and

executing an error handling procedure if it was determined that said writing would overflow a buffer.

22. The method of claim 21, wherein determining whether writing the data string to the block of addresses would overflow a buffer comprises:

concluding that said writing would overflow a buffer whenever both:

a first address within said block of addresses is within a currently allocated buffer in the heap; and

the distance from said first address to the address of a last location in said currently allocated buffer is less than the size of the data string.

23. The method of claim 22, wherein the fault-containment wrapper also keeps track of the allocation of buffers in the heap, and wherein information about the buffer allocation is stored in a buffer allocation data structure.

24. The method of claim 23, wherein the method further comprises determining whether the first address is within said block of addresses within a currently allocated buffer based on information from the buffer allocation data structure.

25. The method of claim 24, the method further comprises:

searching the memory linearly from said first address within said block of addresses for a buffer marker;

confirming whether an actual buffer marker has been found in said search by using information in said buffer allocation data structure; and

concluding that the first address within said block of addresses is not within a currently allocated buffer if the linear search reaches a boundary of the heap without confirming that a buffer marker has been found.

26. The method of claim 25, wherein the search for a valid buffer marker assumes that each memory buffer is aligned at a boundary that is proportional to its size.

27. A fault-containment wrapper for detecting heap buffer overflow, the fault-containment wrapper comprising instructions to:

determine whether writing the data string to the block of addresses would overflow a buffer within the heap;

cause the data string to be written to the block of addresses if it was determined that said writing would not overflow a buffer; and

execute an error handling procedure if it was determined that said writing would overflow a buffer.

28. The fault-containment wrapper of claim 27, wherein the fault-containment wrapper further comprises instructions to conclude that said writing would overflow a buffer whenever both:

a first address within said block of addresses is within a currently allocated buffer in the heap; and

the distance from said first address to the address of a last location in said currently allocated buffer is less than the size of the data string.

29. The fault-containment wrapper of claim 27, wherein the fault-containment wrapper also keeps track of the allocation of buffers in the heap, and wherein information about the buffer allocation is stored in a buffer allocation data structure.

30. The fault-containment wrapper of claim 27, wherein the fault-containment wrapper further comprises instructions to:

search the memory linearly from said first address within said block of addresses for a buffer marker;

confirm whether an actual buffer marker has been found in said memory being searched by using information in said buffer allocation data structure; and

conclude that the first address within said block of addresses is not within a currently allocated buffer if the linear search reaches a boundary of the heap without confirming that a buffer marker has been found.

31. The fault-containment wrapper of claim 30, wherein the search for a valid buffer marker assumes that each memory buffer is aligned at a boundary that is proportional to its size.